Vol.2014-DPS-158 No.37
Vol.2014-CSEC-64 No.37
2014/3/7

デバイスペアリングを利用した位置証明プロトコルの提案

小田 将之[1,a)]　橋本 健二[2]　楫 勇一[1]　関 浩之[2,1]

概要：近年，利用者の位置に応じた情報を提供する位置情報サービスが注目されている．しかし，位置情報の改竄によりサービス提供者や他のユーザが不利益を被ることがある．また，位置情報はユーザのプライバシー情報であるため，その保護についても十分配慮しなければならない．このような理由から，ユーザの位置情報が正しいことをユーザのプライバシーをできるだけ開示せずに証明するための位置証明プロトコルが提案されてきたが，改竄防止等のため位置証明書生成に関わる参加者全てに PKI 等の基盤を仮定しており運用面からは必ずしも簡便とはいえない．本研究では，位置証明プロトコルとデバイスペアリングを組み合わせたプロトコルを提案し，従来手法と比較してプロトコルの軽量化が可能であることを示す．また，提案プロトコルの安全性やプライバシーなどの性質について議論する．さらに，我々が既に開発済みの，人間の動作に基づく共通鍵生成法をデバイスペアリングとして利用し，Android モバイルデバイス上で動作するアプリケーションとして提案手法を実装した．実証実験の結果，正当なユーザに位置証明書を発行できる確率は 93.2%，正当でないユーザに発行してしまう確率は 0%であった．また，位置証明書の生成と検証にかかる実行時間はそれぞれ平均 37 ミリ秒，7 ミリ秒であった．

キーワード：プライバシー，位置情報，デバイスペアリング，加速度センサ，モバイル

1. はじめに

私達には「いつ」「どこに」居たかの証明，すなわち位置証明やアリバイ証明が必要な場面が多々ある．例えば，海外出張から帰国した際に，パスポート等の提出が求められることがある．このような手続きは煩雑であるばかりでなく，それらの書類にはプライベートな情報が含まれる．つまり，「位置証明を電子的に行い，かつ，プライバシー情報はできるだけ公開せずに済ませたい」といったニーズが存在する．また，このような話題と関連して，位置情報サービス (Location-Based Services) が挙げられる．位置情報サービスとは，携帯機器の GPS 機能などにより利用者が今いる位置を取得し，それに応じた情報を提供するサービスである．しかし，位置情報は改竄が可能であり，これによりサービス提供者や他のユーザが不利益を被るなどの問題が指摘されている [4]．また位置情報はユーザのプライバシー情報であるため，その保護問題もある [11]．このような理由から，ユーザの位置情報が本当に正しいことをユーザのプライバシーをできるだけ開示せずに証明するためのプロトコル（位置証明プロトコル）が提案されてきた．

位置証明プロトコルにはユーザ，証人，ジャッジという 3 種類の登場人物が必要である．ユーザは位置証明から何らかの利益や恩恵を受ける人であり，証人はユーザの位置証明を証言する人や機関，ジャッジはユーザが位置証明を主張する対象である．位置証明プロトコルの目的は，ユーザが何者であるかの情報 (Identity) を信頼できる第三者であるジャッジ以外には公開せず，ユーザが証人の助力を得て位置証明書を生成し，ジャッジに対してその証明書の正当性を認めさせることである．しかし，従来研究では，改竄防止等のためユーザと証人の両方に PKI 等の基盤を仮定しており運用面からは必ずしも簡便とはいえず，より軽量なプロトコルが望まれていた．

本研究では，デバイスペアリングを利用することで，ユーザへの PKI の仮定を省略し，位置証明プロトコルの軽量化を行った．デバイスペアリングとは，2 台のデバイスの接続設定であり，特に 2 台の間で安全な通信を行うための鍵共有を行うことである．提案する位置証明プロトコルで利用するデバイスペアリングは，以下の 2 つの条件を満たす必要がある．

- ユーザと証人が特定の時刻に特定の場所に一緒に居たということ（同一のコンテキスト）が保証される．
- ユーザの Identity を証人に公開せずに済む．

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[2] 名古屋大学
Nagoya University
[a)] masayuki-o@is.naist.jp

Vol.2014-DPS-158 No.37
Vol.2014-CSEC-64 No.37
2014/3/7

例えば，RSA による単なる鍵共有では，ユーザと証人が同一のコンテキストであることが保証されず，用いることができない．また，ユーザが電子署名を用いれば，その検証鍵が PKI によってユーザの Identity に紐付いてしまい，上に挙げた 2 つの条件を満たさない．そこで，2 つの条件を満たすペアリング手法として，手軽かつ安全性の高い加速度センサを用いた鍵自動生成法 [7], [8] を採用した．この手法は，2 つのデバイスを重ねて一緒に振る（シェイキング）と，それぞれのデバイスの加速度センサから似たようなセンシングデータが得られるという性質を利用し，その加速度データを解析し加工することで同一の鍵を生成，すなわち鍵共有を行う．したがって，ユーザと証人それぞれのデバイスを重ねて一緒に振ることでユーザと証人しか持ち得ない情報（鍵）が共有され，一緒に振らなければ鍵が共有されないため，同一のコンテキストであることが証明可能となる．また，理論的にはユーザの Identity も非公開のまま鍵共有が可能である．以上の理由と，鍵の生成効率が優れるという点で，このペアリング手法を提案する位置証明プロトコルに適用した．

本論文では，以上の提案する位置証明プロトコルについて，脅威と信頼に関するモデルを設定し，安全性とプライバシー，その他に関する考察を述べる．次に，デバイスペアリングとして我々が既に開発済みの人間の動作に基づく共通鍵生成法を利用し，Android デバイス上に提案手法を実装し実証実験を行ったのでその結果についても述べる．

2. 関連研究

ユーザの位置情報を上手く隠したまま位置情報サービスの恩恵を受けられる枠組みがいくつか提案されている [2], [3], [6]．しかし，これらは位置を証明するという目的ではなく，ユーザの利便性を損なわずにプライバシー情報をできるだけ隠すという目的に使われる枠組みであるため，ユーザの位置情報の正しさは保証しない．例えば，ユーザが偽の位置情報を使ったとしてもそれを検出できるわけではない．

これに対し，ユーザのプライベートな情報を上手く隠したまま位置情報を証明するプロトコルがいくつか提案されており，我々が提案するプロトコルもその中の一つである．ここではこれらの関連研究を紹介する．

2.1 プライバシーを考慮した位置証明プロトコル

[5] では，証人として無線 LAN アクセスポイントが使用される．しかし本研究と大きく異なる点は，証人が永続的に固定されていなければならない点である．我々の提案するプロトコルでは，証人は自由に移動して構わず，どこに居てもユーザから位置証明書発行依頼を受けた時にすぐに位置証明書を発行できる．また [5] では，信頼できる第三者機関が 3 つ必要であり，軽量とはいえない．

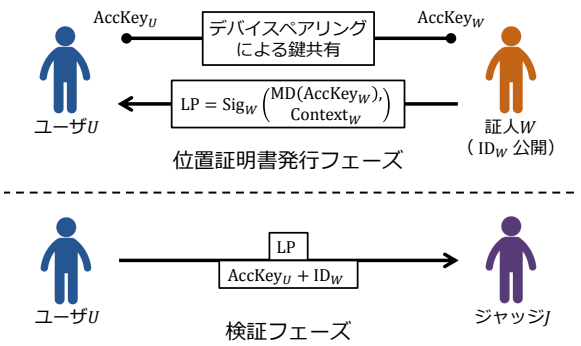

図 1 位置証明プロトコル

[9] は，各ユーザを匿名化することにより位置証明におけるユーザのプライバシーを確保している．しかし，信頼できる第三者機関として認証局（公開鍵証明書を発行する機関）だけでなく，匿名情報とユーザの ID を紐付ける機関が必要である．

[1] は，信頼できる第三者機関が認証局のみで良いという点で 他の 2 つのプロトコルに比べて実用性が高い．このプロトコルが提案手法と異なる点は，ユーザと証人の双方に PKI を要求する点である．

上に挙げた 3 つのプロトコルは，どれも証人側だけでなくユーザ側にも PKI を要求する．しかし，我々の提案するプロトコルでは証人側のみに PKI を要求し，ユーザ側には要求しない．

3. 提案するプロトコル

我々の提案手法には，前提として 3 人の登場人物が存在する．ユーザ U，証人 W，位置証明書を検証するジャッジ J である．プロトコルの目標は，ユーザがプライバシー情報をジャッジ以外の誰にも公開せず，証人の助力によってジャッジに対して位置証明を行うことである．提案プロトコルは，位置証明書発行フェーズと検証フェーズからなる（図 1）．

3.1 初期設定

X のプライバシー情報 i_X を以下のように定義する．

$$i_X = (\mathrm{ID}_X, \mathrm{Context}_X) \tag{1}$$

ここで ID_X とは X が「何者であるか」，$\mathrm{Context}_X = (\mathrm{location}, \mathrm{time})$ は X が「いつ (time)，どこ (location) に居たか」を表す．ユーザと証人は $\mathrm{Context}_X$ を実時間でセンシングするため，$\mathrm{Context}_X$ は変動するが，ID_X は X の氏名や住所などの情報を想定しているため，基本的には固定である．式 (1) より，ID_X や $\mathrm{Context}_X$ のどちらか片方だけではプライバシー情報にはならないことに注意する．プロトコルの目標を達成するためにはユーザのプライバシー情報 $i_U = (\mathrm{ID}_U, \mathrm{Context}_U)$ をジャッジ以外には公開しないようにする必要がある．

提案手法では，証人は，署名鍵と検証鍵のペア (sk_W, pk_W) を所持しており，メッセージ m に対する sk_W による電子署名を $\mathrm{Sig}_{sk_W}(m)$，署名付きメッセージの電子署名 s に対する署名の検証を $\mathrm{Veri}_{pk_W}(s)$ と表記するものとする．以上より，$\mathrm{Veri}_{pk_W}(\mathrm{Sig}_{sk_W}(m)) = m$ が成り立つ．また，前提として PKI によって，ID_W は，W の pk_W と正しく紐付けされており，ID_W は公開されているものとする．現実世界では，本プロトコルの証人に相当するものは駅や空港，役所，学校などの公的機関，あるいは商業上の理由から積極的に ID を公開しようとする企業等を想定している．さらに，m に対する一方向性ハッシュ関数を $\mathrm{MD}(m)$ と表す．

重要な点として，提案手法では電子署名を用いるのは証人のみであり，ユーザは用いない．したがって，ユーザ側には PKI を仮定する必要がない．

3.2 位置証明書発行フェーズ

ユーザは位置証明書を生成したいときに，証人の協力を得て加速度センサを用いた鍵自動生成によるデバイスペアリングを行い，ユーザが AccKey_U，証人が AccKey_W を生成する．これを位置証明書の発行依頼とみなす．デバイスペアリングが正しく行われれば，以下のように 2 つの鍵が等しくなる．

$$\mathrm{AccKey}_U = \mathrm{AccKey}_W \tag{2}$$

鍵生成と同時に，ユーザと証人はそれぞれ $\mathrm{Context}_U$ と $\mathrm{Context}_W$ も作成する．それらはほぼ同じ環境で生成される情報であるため非常に近い値となるはずである．本プロトコルで用いるデバイスペアリングは，3.3 節で示すように，この特性が重要である．また，AccKey はユーザと証人の間だけの秘密にしておかなければならないが，証人はこれを保存しておく必要はない．

次に，証人は以下の計算を行う．

$$E = (\mathrm{MD}(\mathrm{AccKey}_W), \mathrm{Context}_W) \tag{3}$$

これに，証人による電子署名を添えたものを位置証明書 LP とし，ユーザに送り返す．

$$\mathrm{LP} = (E, \mathrm{Sig}_{sk_W}(E)) \tag{4}$$

このとき，安全な通信路を仮定しなくてもよい（詳細は 4.2.1 および 4.2.2 節）．また，証人は位置証明用に生成した全てのデータを破棄してよい（詳細は 4.2.3 節）．

ユーザは，受け取った位置証明書 LP が，正しい位置証明書であることを以下の計算によって確認できる．

$$\mathrm{Veri}_{pk_W}(\mathrm{Sig}_{sk_W}(E)) = E \tag{5}$$

$$\mathrm{MD}(\mathrm{AccKey}_U) = \mathrm{MD}(\mathrm{AccKey}_W) \tag{6}$$

$$\mathrm{Context}_U \approx \mathrm{Context}_W \tag{7}$$

pk_W は，ID_W が公開されていることと，PKI により取得可能である．また，“≈” は，非常に近い値であることを示す．ここで，式 (5) により LP が改竄されていないこと，式 (6) によりデバイスペアリングが正しく行われたこと，式 (7) により LP が虚偽のコンテキスト情報でないかを検証できる．どれか一つの式でも成り立たなければ，LP の生成は失敗となり，ユーザは受け取ったデータを破棄する．全ての式が成り立てば，ユーザは LP と $\mathrm{AccKey}_U, \mathrm{ID}_W$ を保存しておき，位置証明を行いたいときにこれらをジャッジに掲示する．

位置証明書発行フェーズで重要な点は，ユーザのみがこれらのデータを保存するが，証人にはデータの保存を一切要求しない点である．すなわち，ストレージ面で証人の負担はない．

3.3 検証フェーズ

ユーザは位置証明を行いたいときに，ジャッジに対して LP と $\mathrm{AccKey}_U, \mathrm{ID}_W, \mathrm{ID}_U$ を掲示する．ジャッジは ID_W を元に PKI から pk_W を取得し，式 (5) と式 (6) を検証する．前者は，LP が証人本人のものかどうかの署名検証，後者は，LP がユーザに対して送られたものかどうかの検証である．両式が成り立てば，$\mathrm{Context}_U \approx \mathrm{Context}_W$ であるといえる．以上より，ID_U と LP に含まれる $\mathrm{Context}_W$ が結びつき，ユーザの位置証明が完了となる．

検証フェーズで重要な点は，ユーザがジャッジに対して位置証明を行う際，証人の手を借りる必要がないことである．証人は位置証明書発行フェーズでのみユーザに手を貸すだけでよく，ストレージ面はもちろん，時間的にも証人の負担が少ない．

4. プロトコルの性質に関する議論

4.1 脅威モデルと信頼モデル

本論文では，以下の 5 種類の脅威について検討する．

- 正直でないユーザ：実際には，特定の時刻に特定の場所に居なかったにも関わらず，偽の位置証明書を生成して，虚偽の位置証明を成立させようとする．
- 悪意のある証人：ユーザに偽の位置証明書を発行しようとする．あるいは，ユーザの依頼で生成した位置証明書を使い，ユーザになりすまして位置証明を行おうとする．
- 証人のなりすまし：実際は正しい証人ではないにも関わらず，位置証明書を発行しようとする．
- 受動的および能動的な盗聴者：ユーザと証人の間での通信を記録したり，改竄しようとする．
- ユーザに協力する第三者：位置証明には興味がないが，ユーザの代わりに位置証明書を取りに行き，受け取った位置証明書をユーザに渡すことで金銭等のインセンティブを得る．

ジャッジは信頼できると仮定する．ユーザが証人の助力を得て位置証明書を生成する際は，安全な通信路を仮定しないが，ユーザがジャッジに位置証明書を掲示する際は，TLS/SSL による通信もしくは直接渡す等のセキュアな通信路を用いる．したがって，盗聴は位置証明書発行フェーズでは起こりうるが，検証フェーズでは起こりえないと仮定する．また，本プロトコルで用いる鍵 $(sk_w, pk_w, \mathrm{AccKey})$ と一方向性ハッシュ関数は，計算量的に十分安全であると仮定する．

攻撃者がユーザの ID を知る機会は，提案するプロトコルにおけるメッセージの通信中のみにしかないと仮定する．

証人のサービス拒否攻撃は考慮しない．

ユーザと証人，あるいは，ユーザとユーザに協力する第三者など 2 人以上が結託すると，検証に成功する偽の位置証明書（偽のアリバイ）が生成できてしまう．これは，現実の世界と同様回避は困難であるが，登場人物中の誰か 1 人が一方的に偽の位置証明書を作ろうとするのは検知できる．この点は，現実の世界におけるアリバイとは異なる．

4.2 プロトコルの性質

4.2.1 安全性

位置証明書 LP は，ユーザの AccKey，証人の ID，そして両者のコンテキストに紐付けられている．ここでは，ユーザ，もしくは証人どちらかが一方的に偽の LP を作ろうとしても不可能であることを示す．

まず，ユーザと証人，盗聴者の誰か 1 人が勝手にユーザの位置証明書を改竄できるかどうかについて議論する．

- ユーザが位置証明書を受け取った後で，位置証明書を改竄するのは不可能である．なぜなら，位置証明書には証人の署名が入っているために改竄が検知でき，検証フェーズで失敗するからである．
- 証人が位置証明書を生成するとき，偽の位置証明書を生成するのは不可能である．なぜなら，ユーザが位置証明書を受け取るときに，式 (5)(6)(7) が成り立つことを確かめることにより，位置証明書が正しいかどうかを検証できるからである．
- 証人がユーザに位置証明書を送っているときに，盗聴者が位置証明書を改竄するのは不可能である．なぜなら，位置証明書には証人の署名が入っているためユーザが改竄を検知するからである．

次に，位置証明書を誰かが配付したり，ユーザの代わりに位置証明書を使うことができないかを議論する．

- ユーザは証人から受け取った位置証明書を第三者に配付する可能性があるが，位置証明書単体では有効ではない．なぜなら AccKey も一緒に配付しなければ，有効な位置証明書とはならないからである．ただし，位置証明書に AccKey を添えて配付した場合，有効な位置証明書を複数の人が入手できてしまうが，ジャッジが同一の位置証明書を受け付けないようにすることで，同一の位置証明書の乱用を防ぐことは可能である．
- 証人は位置証明書に AccKey を添えて配付することができてしまうが，位置証明書はユーザにすでに渡っているので，ユーザの位置証明が無効になるわけではない．また，上に述べたようにジャッジが同一の位置証明書を受け付けないようになっている場合，ユーザの位置証明書が使えなくなる可能性があるが，それと同時に証人は社会的信用を失うリスクを負う．
- 証人がユーザの依頼を受けて生成した位置証明書を使って，もしくはデバイスペアリングを行わず証人のみで位置証明書を生成して，位置証明を行おうとしても失敗する．なぜなら，署名が証人自身によるものであり，それは結局位置証明書の証人を自分自身が行っていることと同義であるため，ジャッジはその事実を検出できるからである．
- 盗聴者は，位置証明を配付したり，自分で使おうとしても有効ではない．なぜなら，AccKey を入手できないため，検証フェーズで失敗するからである．

最後に，誰かがユーザや証人になりすますことができないかを議論する．

- ユーザ以外のいかなる人（盗聴者，証人を含む）も，ユーザになりすまして位置証明を行うことはできない．なぜなら，誰も位置証明書を受け取ったユーザの ID が分からないからである．
- ユーザも含め誰も証人になりすますことができない．なぜなら証人の署名鍵を持っていないからである．

以上の議論により，正直でないユーザ，悪意のある証人，証人のなりすまし，および盗聴者の不正は検出できることが示された．一方，ユーザに協力する第三者およびユーザと証人の結託は防ぐことができない．しかし，この問題は現実の世界でも同様である．

4.2.2 プライバシー

ここでは，ユーザのプライバシーが保たれているかどうかを議論する．

- ユーザが位置証明書をジャッジに提出し，位置証明を主張するまで誰であろうと（証人を含む）ユーザの ID を知ることはできない．なぜなら，位置証明書を生成する際に使う情報にも，位置証明書の中身にもユーザ ID の情報は含まれていないからである．
- ジャッジに対して位置証明を行うときは，ユーザは自身の ID をジャッジに示す必要があるが，証人をその場に呼び出す必要がないため，証人やその他の第三者にユーザの ID が知られることはない．

4.2.3 効率

ストレージはユーザのみが必要で，証人はいかなるデータも保存しておく必要がない（もちろん証人自身の署名鍵はもっておく必要がある）．この点は，ネットワークを構

Vol.2014-DPS-158 No.37
Vol.2014-CSEC-64 No.37
2014/3/7

表 1　Nexus 7 (2013) の仕様

OS	Android 4.3 Jelly Bean	
CPU	Krait @ 1.50GHz	
サイズ	200 × 114 × 8.65mm	
重量	290g	
その他	加速度センサ，NFC，GPS 対応	
加速度センサの仕様	検出軸数	3 軸
	最大サンプリングレート	200Hz

築するための鍵共有とは大きく異なり，証人にはストレージ面での負担がかからない．一方，ユーザはストレージを用意しなければならないが，ユーザには自身の位置証明が行えるというインセンティブがあるため，この負担は合理的である．

5. 実装と評価

5.1 実験環境

本プロトコルの実用性を評価するために，Android モバイルデバイスでの実装を行った．今回は，Nexus 7 (2013) で性能評価と実行時間の計測を行った（**表 1**）．

電子署名のアルゴリズムに 2048 bit の RSA 暗号，一方向性ハッシュ関数には，SHA-256 を使用した．ユーザと証人との間のデータ交換には NFC (Near Field Communication) を用いている．NFC は一般に安全な通信路とはいえないが，すでに 4 章で議論した通り，盗聴や改竄への対策がなされているため問題にならない．また，位置情報の取得には GPS を利用している．

5.2 デバイスペアリングの実装と評価

5.2.1 加速度センサを用いた鍵自動生成の実装

デバイスペアリングのプロトコルは，[10] で使用されたパッケージを Android での実装向けにカスタマイズして用い，[7], [8] で提案されている加速度センサのセンシングデータを周波数領域で解析，鍵生成を行う手法をとった．Android モバイルデバイスで加速度センサを用いた鍵自動生成を行うには，**図 2** のように 2 つのデバイスを重ねてシェイキングを行う．具体的なパラメータは，**表 2** のように設定した．これにより生成される鍵の長さは最大で 1 秒あたり約 94.5 bit である．

プログラムは，[10] で作成されたパッケージを利用し，

表 2　鍵生成のパラメータ設定

パラメータ	値
サンプリングレート	200Hz
W_{fft} （FFT の窓サイズ）	128
W_{fft} のオーバーラップ	50%
b （量子バンド数）	6
c （鍵小片の候補数）	6
α （境界の増加幅）	0.5
$f^q_{\max}$ （量子化する周波数帯最大値）	20Hz

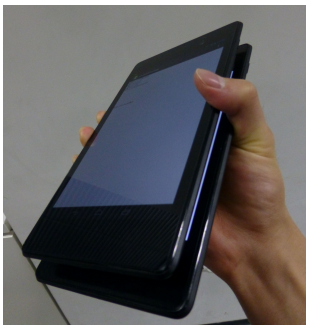

図 2　デバイスペアリングを行う際の 2 つのデバイスの重ね方

Android デバイス向けにカスタマイズを行った．[10] では 2 つの加速度センサを 1 つの PC に接続することが前提であったため，PC の時間情報を用いて同期していたが，今回は 2 つのデバイス各々でセンシングするため，2 つのデバイスを同時に強く振った時点を 0 秒としてカウントすることを試みた．またそのためのノイズを低減させる IIR フィルタとして，センシング開始時刻 $t = 0$ から現在時刻 $t = i$ までのセンシングデータ a_i の指数移動平均 b_i を以下に従って計算する ($\beta = 0.8$)．

$$\begin{aligned} b_i &= (1-\beta)\sum_{j=0}^{i}\beta^j a_{i-j} \\ &= (1-\beta)a_i + \beta b_{i-1} \end{aligned} \tag{8}$$

このフィルタによりデバイスを振っていない状態で時間のカウントが開始されることはなくなり，実験でも優れた結果が得られている（5.2.2 節）．

強く振るという動作を判定する閾値については，予備実験の結果，$b_i = 18$ [m/s^2] を超えた時点でカウントが開始されるように実装した．シェイキングの終了は，カウント開始から加速度値を 1024 サンプル（約 5 秒）取得するまでと設定したため，生成される鍵小片の数は 15 個となる．

5.2.2 加速度センサを用いた鍵自動生成の実験と評価

加速度センサを用いた鍵自動生成の性能を評価するために，2 つの実験を行った．1 つ目の実験は，false negative（2 つのデバイスを重ねて同時にシェイキングしたにも関わらず同一の鍵が生成されない）を評価するための実験である．1 人の被験者が 2 つのデバイスを重ねて持ち，シェイキングを行い，15 個生成される鍵小片がいくつ一致したかを記録する．13 名の被験者に合計 59 回約 5 秒（1024 サンプル）のシェイキングを行ってもらった．

2 つ目の実験は，false positive（2 つのデバイスを重ねて同時にシェイキングしていないにも関わらず誤って同一の鍵が生成されてしまう）を評価するための実験である．被験者はペアを組み，それぞれが 1 つずつデバイスを片手で持ち，どちらかがもう一方のシェイキング動作を真似することで鍵生成を行う．被験者ペアにとっての目標はお互いにできる限り類似したシェイキングパターンを生成するこ

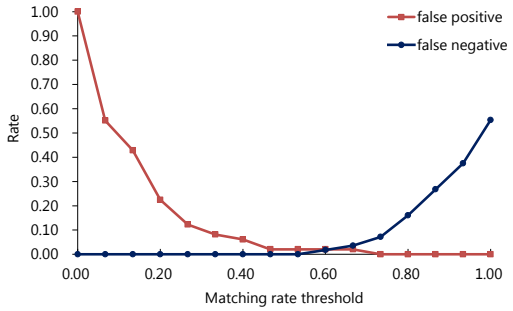

図 3 鍵小片の一致率の閾値と false positive/negative の関係

表 3 各操作の平均実行時間

操作	実行時間（ミリ秒）
署名鍵と検証鍵のペア生成	2566
位置証明書の生成	37
位置証明書の検証	6

とである．被験者ペアのインセンティブになるように，15 個の鍵小片のうち 8 個（約 53.3%）以上が一致した場合は景品を授与するルールを設けた．被験者 14 名 8 ペア（重複含む）に合計 54 回のシェイキングを行ってもらった．

実験結果から得られた鍵小片の一致率ごとの false positive/negative の割合を**図 3** に示す．鍵小片の一致率が 73.3%（15 個中 11 個）以上のときを鍵生成の成功とし，それに満たない場合は鍵生成の失敗としたとき，false positive が 0%となり，そのときの false negative は 6.8%という結果となった．

5.3 位置証明プロトコルの実装と評価

位置証明アプリケーションが現実の世界で実用的な時間で利用可能かどうかを調べるために，位置証明プロトコルの各操作にかかる実行時間を計測した．証人が署名鍵と検証鍵の鍵ペアを生成するのにかかる時間，位置証明書の生成にかかる時間，および位置証明書の検証にかかる時間の 3 つである．位置証明書の検証とは，式 (5)(6)(7) が全て成り立つかの検証である．計測は 1000 回実行した平均値を取った．結果を**表 3** に示す．署名鍵と検証鍵のペア生成に平均 2566 ミリ秒の時間がかかるものの，これは前処理であり一度しか行う必要がない．位置証明書を生成する時間は平均 37 ミリ秒，位置証明書の検証には平均 6 ミリ秒の時間で実行できた．以上より，提案する位置証明プロトコルが実用的な精度と時間で実行できることが分かった．

6. おわりに

本研究では，従来のプライバシーを考慮した位置証明プロトコルを軽量化することを目標として，デバイスペアリングを利用することで，ユーザに PKI を用いることを要求しない位置証明プロトコルを提案し，Android モバイルデバイス上で実装と評価を行った．

提案するプロトコルで用いるデバイスペアリングは以下の要件を満たす必要がある．

- ペアリングを行う時点でのユーザと証人のコンテキストが（ほぼ）一致する．
- ユーザの ID を証人に公開せずに済む．

この要件を満たすデバイスペアリングとして，我々の研究グループが既に開発済みの，人間の動作に基づく共通鍵生成法を利用し，Android デバイス上に提案手法を実装し実証実験を行ったところ，デバイスペアリングでは，同時に振った 2 台のデバイスが正しく同一の鍵を生成できる確率は 93.2%であり，同時に振っていない 2 台のデバイスが同一の鍵を誤って生成する確率は 0%であった．また，位置証明プロトコルは，証人の前処理として署名鍵と検証鍵のペア生成にかかる時間が平均 2566 ミリ秒かかるものの，位置証明書を生成する時間は 37 ミリ秒，位置証明書の検証には 6 ミリ秒の時間で実行できた．このように，提案する位置証明プロトコルが実用的な精度と時間で実行できることが分かった．提案プロトコルの拡張，応用としては，位置証明書にコンテキスト以外（例えば，領収書）の情報を埋めこむことで，コンテキスト以外の情報を証明するプロトコルの設計が可能である．

参考文献

[1] B. Davis, H. Chen, and M. Franklin. Privacy-preserving alibi systems. *ACM ASIACCS*, 2012.
[2] M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. *Pervasive*, pp. 152–170, 2005.
[3] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan. Private queries in location based services: Anonymizers are not necessary. *ACM SIGMOD*, pp. 121–132, 2008.
[4] W. He, X. Liu, and M. Ren. Location cheating: A security challenge to location-based social network services. *IEEE ICDCS*, pp. 740–749, 2011.
[5] W. Luo and U. Hengartner. VeriPlace: A privacy-aware location proof architecture. *ACM SIGSPATIAL GIS*, pp. 23–32, 2010.
[6] C. Mascetti, X. Wang, and S. Jajodia. Anonymity in location-based services: Towards a general framework. *MDM*, pp. 69–76, 2007.
[7] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. *Pervasive*, pp. 144–161, 2007.
[8] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE TMC*, vol. 8, no. 6, pp. 792–806, 2009.
[9] Z. Zhu and G. Cao. APPLAUS: A privacy-preserving location proof updating system for location-based services. *IEEE INFOCOM*, pp. 1889–1897, 2011.
[10] 南貴博, 仁野裕一, 野田潤, 中村嘉隆, 関浩之. ユーザの動作類似度に基づく共通鍵生成法. 情報処理学会研究報告, No. 2009-CSEC-44, 2009.
[11] 日本経済新聞. 「ロケハラ」にご注意、位置情報サービスの落とし穴 スマホ普及でプライバシー侵害と背中合わせ. `http://www.nikkei.com/article/DGXBZO34976640R20C11A9000000/`, 2011. 2014 年 2 月 2 日閲覧.